



Europäisches Patentamt  
European Patent Office  
Office européen des brevets



Publication number: **0 669 586 A2**

(12)

## EUROPEAN PATENT APPLICATION

(21) Application number: 95102556.8

(51) Int. Cl. 6: G06F 17/60

(22) Date of filing: 23.02.95

(30) Priority: 25.02.94 US 201664

(43) Date of publication of application:  
30.08.95 Bulletin 95/35

(84) Designated Contracting States:  
DE FR GB IT

(71) Applicant: MINNESOTA MINING AND  
MANUFACTURING COMPANY  
3M Center,  
P.O. Box 33427  
St. Paul,  
Minnesota 55133-3427 (US)

(72) Inventor: Sisley, Elizabeth M., c/o Minnesota  
Mining and  
Manuf. Co.,  
2501 Hudson Road,  
P.O. Box 33427  
Saint Paul,  
Minnesota 55133-3427 (US)  
Inventor: Collins, John E., c/o Minnesota  
Mining and  
Manuf. Co.,  
2501 Hudson Road,  
P.O. Box 33427  
Saint Paul,  
Minnesota 55133-3427 (US)

(74) Representative: Hilleringmann, Jochen,  
Dipl.-Ing. et al  
Patentanwälte  
von Kreisler-Selting-Werner,  
Bahnhofsvorplatz 1 (Deichmannhaus)  
D-50667 Köln (DE)

(54) System and method for resource assignment and scheduling.

(57) A system and method for assigning and scheduling resource requests to resource providers use a modified "best-first" search technique that combines optimization, artificial intelligence, and constraint-processing to arrive at near-optimal assignment and scheduling solutions. In response to changes in a dynamic resource environment, potential changes to an existing assignment set are evaluated in a search for a better solution. New calls are assigned and scheduled as they are received, and the assignment set is readjusted as the field service environment changes, resulting in global optimization. Each search operation is in response to either an incremental change to the assignment set such as adding a new resource request, removing a pending resource request, reassigning a pending resource request, or to a request for further evaluation. Thus, the search technique assumes that the existing assignment set is already optimized, and limits the task only to evaluating the effects of the incremental change. In addition, each search operation produces a complete assignment and scheduling solution. Consequently, the search can be terminated to accept the best solution generated so far, making the technique an "anytime" search.

EP 0 669 586 A2

Field of the Invention

The present invention relates to resource management, and, more particularly, to techniques for the assignment and scheduling of resource requests among a group of resource providers.

Discussion of Related Art

Resource assignment and scheduling problems exist in many organizations such as, for example, those involved in service or manufacturing activities. In the abstract, the assignment and scheduling problem requires the assignment of resource requests to appropriate resource providers, and the scheduling of the resource requests at particular times. Techniques for solving the assignment and scheduling problem must consider various factors including the types of resources required by particular resource requests, the types of resources available from individual resource providers, previous assignments of pending resource requests, and the scheduled times and durations of the pending resource requests.

One example of an assignment and scheduling problem arises in the context of a field service environment. A field service environment exists in many organizations, characterized by a group of field service technicians dedicated to the repair and maintenance of a variety of industrial machines, office equipment, and the like. The field service technician travels to the customer's location to perform preventative maintenance and to provide repair services pursuant to customer service calls. Thus, in the field service environment, the technicians function as resource providers, providing maintenance and repair services, and both customer service calls and preventative maintenance appointments serve as resource requests.

When a customer calls to report a problem, a human service call taker creates a call order in a database software application called a Service Management System (SMS). The SMS also may generate call orders automatically for periodic preventative maintenance appointments. The call taker adds details to the call order concerning the customer, the machine, the contract, and any symptoms the customer can provide, and then transmits the call information to a human service call dispatcher via a computer terminal or a printed call order slip. The service call dispatcher is then responsible for assigning the service call to an individual service technician, and scheduling the call among the other calls assigned to the technician. To issue assignment and scheduling decisions, the human service call dispatcher relies on either individual judgment, or one of a number of automated assignment and scheduling techniques.

In the absence of automated techniques, the human dispatcher manually posts slips of paper on a large magnetic board to indicate present assignment and scheduling decisions. With reference to the magnetic board, the dispatcher determines which technicians should be assigned a new service call based on various considerations including the characteristics of the call, the customer's contract requirements, the dynamic call load among the technicians, and available technician resources. The dispatcher places the new service call in an area of the magnetic board marked "pending" for a particular technician who the dispatcher considers most likely to take the call. When the technician telephones the dispatching center, the dispatcher simply views the magnetic board and reads one or more calls from the "pending" area. If the technician agrees to accept assignment of the calls, the dispatcher moves them to an area marked "committed."

Unfortunately, the manual dispatching technique suffers from several disadvantages. First, because the pace of the dispatcher's work varies from slow to hectic, usually with a bimodal distribution, the dispatcher's planning tends to be poorest during periods of hectic activity. Although the efficiency of the assignments made during this time suffers, the dispatcher is unlikely to reevaluate the assignments and modify them later during a slow interval. Second, the dispatchers tend to make most decisions based on a small set of heuristics. The combinatorics of the planning problem limit the extent of the human dispatcher's evaluation, effectively preventing the dispatcher from consistently finding the best solutions. Third, the only way to get an overview of the field situation in terms of loading of technicians and commitments made to customers is to look at the board in the dispatching center. Consequently, this information is not readily available to the technicians and managers in the field. Without such information, the technicians and managers are unable to contribute any valuable input to the dispatcher's planning problem.

A number of automated assignment and scheduling techniques have been developed to improve or replace the decisionmaking function of the human dispatcher. Existing automated assignment and scheduling techniques include, for example, the "primary technician" approach of automatically assigning service calls to technicians designated as primary technicians for particular customer accounts, the "one-call-at-a-time" approach of automatically assigning one of a plurality of new calls to an available technician based on an objective function, and a pure mathematical optimization approach of evaluating virtually all feasible assignment and scheduling combinations for a predetermined set of calls according to an objective function.

ment and scheduling solutions, while avoiding the combinatorial explosion inherent in existing optimization techniques. In particular, the present invention provides a system and method for assigning a plurality of resource requests among a plurality of resource providers, wherein the plurality of resource requests includes a plurality of pending resource requests assigned among the resource providers according to an existing assignment set, and wherein the existing assignment set defines a root node of a search tree. According to the system and method of the present invention, the root node is expanded by forming one or more next-level nodes, each of the next-level nodes corresponding to the root node but being further defined by a reassignment of one of the pending resource requests between one of the resource providers and another of the resource providers. A stress value is estimated for each of the next-level nodes, wherein the stress value represents a degree of undesirability of the respective reassignment, and a new assignment set is generated corresponding to one of the next-level nodes having a minimum stress value.

It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only, and not restrictive of the invention, as claimed.

The accompanying drawings are included to provide a further understanding of the invention and are incorporated in and constitute a part of this specification. The drawings illustrate exemplary embodiments of the invention and together with the description serve to explain the principles of the invention.

#### **Brief Description of the Drawings**

- Fig. 1 is a functional block diagram of a computer-implemented software process structure incorporating a system for assigning and scheduling resource requests in accordance with the present invention;
- Fig. 2 is a functional block diagram of a system for assigning and scheduling resource requests in accordance with the present invention;
- Fig. 3 is a flow diagram illustrating a first mode of a system and method for assigning and scheduling resource requests in accordance with the present invention; and
- Fig. 4 is a search tree diagram illustrating the operation of the first mode shown in Fig. 3;
- Fig. 5 is a flow diagram illustrating a second mode of a system and method for assigning and scheduling resource requests in accordance with the present invention;
- Fig. 6 is a flow diagram illustrating a scheduling mode of a system and method for assigning and scheduling resource requests in accordance with the present invention;
- Fig. 7 is a scheduling diagram illustrating a lower bound calculation performed in the scheduling mode of a system and method for assigning and scheduling resource requests in accordance with the present invention;
- Fig. 8 is a search tree diagram illustrating a schedule sequence of resource requests for a particular resource provider in accordance with the present invention;
- Fig. 9 is a scheduling calendar illustrating the scheduling of resource requests in accordance with the present invention; and
- Fig. 10 is an example of a user interface displaying a set of schedules generated in accordance with the present invention.

#### **Detailed Description of the Preferred Embodiments**

Reference will now be made in detail to exemplary embodiments of the invention. One skilled in the art, given the description herein, will recognize the utility of the system and method of the present invention in a variety of diverse resource environments in which assignment and scheduling problems exist. For example, it is conceivable that the system and method of the present invention may be adapted for assignment and scheduling problems existent in telecommunications systems, transportation dispatching organizations, and emergency services dispatching organizations. However, for ease of description, as well as for purposes of illustration, the present invention will be described in the context of a field service environment.

The system and method of the present invention will be described together herein, with the method contemplated as being implemented as a series of operations performed by the system. Fig. 1 is a functional block diagram of a computer-implemented software process structure 10 configured for application in a field service environment as described above. The software process structure 10 incorporates a system 12 for the assignment and scheduling of service calls in accordance with the present invention. Assignment/scheduling (A/S) system 12 is a software system realized, for example, by a software process running on a standard Unix™ workstation. The A/S system 12, which may be implemented using the Common Lisp Object System (CLOS), is designed to run as a back-end processor for an existing SMS (not shown). The software process structure 10 further comprises an SMS interface 14 having an SMS access

The structure of A/S system 12 will now be described with reference to Fig. 2. The A/S system 12 employs two software process modules that cooperate to reach an assignment set that optimizes a single objective function of the field service environment. As shown in Fig. 2, A/S system 12 comprises an assigner module 22 and a scheduler module 24. The assigner module 22 searches for potential assignments of service calls among the service technicians, and evaluates a portion of the objective function relating to the desirability of particular associations of calls and technicians. The assigner module 22 invokes scheduler module 24 to search for the best schedule among a plurality of potential schedules of the calls assigned to a particular technician, and to evaluate a portion of the objective function relating to time. Each of the potential schedules searched by scheduler module 24 represents a time-ordered sequence of the calls assigned to a technician. Thus, a complete assignment of a service call involves both an association of the call with a technician, as determined by assigner module 22, and a scheduling of the call at a particular time, as determined by the scheduler module 24.

The A/S system 12 further includes three main data structures 26, 28, 30, and a clock 32 accessed by assigner module 22. The assignment set data structure 26 stores a representation of the existing assignment set that defines an existing assignment of pending service calls among the service technicians. The assignment set data structure 26 also includes an existing schedule set that defines a schedule of the pending service calls assigned to each of the service technicians. The technician set data structure 28 stores a representation of the existing technician set that defines the attributes of the technicians operating in the field service environment. The call set data structure 30 stores a representation of the existing call set that defines the attributes of the pending service calls. The clock 32 serves as a source of real-time for use in evaluating time-related factors of an assignment. The assigner module 22 continuously updates the technician and call data structures 28, 30 to reflect changes in the field service environment as represented by the SMS and field events received by queue 20. The assigner module 22 also updates the assignment set stored in the assignment set data structure 26 to represent assignment and scheduling recommendations.

As shown in Fig. 2, the scheduler module 24 accesses a calendar data structure 34 during the scheduling operation. The calendar data structure 34 stores a representation of the working calendar of the organization operating in the field service environment. The working calendar indicates time segments potentially available for the scheduling of service calls, and may be updated according to organizational calendar changes via user interfaces 18. Each of the time segments may correspond to the length of a work day. Thus, the scheduler module 24 accesses the calendar data structure 32 to determine which time segments are available work days, and to determine scheduling boundaries between consecutive work days to avoid carrying calls over to the next day.

The search operation of A/S system 12 will now be described. In response to an SMS or field event received at queue 20, the A/S system 12 conducts a modified "best-first" search that combines optimization, artificial intelligence, and constraint-processing techniques to arrive at near-optimal assignment and scheduling recommendations. When a change to the technician or call set is received, the A/S system 12 executes a search that evaluates potential changes to the existing assignment set in an attempt to find a better distribution of calls among the technicians, as well as a better scheduling of calls assigned to individual technicians. The A/S system 12 generates assignment and scheduling recommendations for all new calls as they are received, and immediately readjusts the assignment, resulting in global optimization. However, each search operation conducted by A/S system 12 explores the effects of only an incremental change to the call set such as adding a new call, removing a pending call, or reassigning a pending call. Thus, the search technique assumes that the existing assignment set is already optimized, and limits the task only to minimizing the effects of the incremental change, thereby reducing the search space. The A/S system 12 also incorporates domain constraints into the search to further prune the search space.

With each search operation, the A/S system 12 produces a complete solution for the field service environment, i.e., all calls are assigned to technicians. Consequently, the search can be stopped at any time, taking the best solution found so far. This characteristic makes the search technique conducted by A/S system 12 an "anytime search," and enables the search to be suspended for other activity, but restarted later for improvements on the existing solution. For example, the A/S system 12 may search for better solutions in response to either a request for reevaluation by a system user, the passage of a predetermined amount of time, or simply the availability of idle processing time.

The SMS and field events received by queue 20 can be divided into the following three basic categories: a change to the call set, a change to the technician set, and a request for further evaluation. The response of the A/S system 12 to the three categories of events falls into two basic modes. The addition of a new service call initiates the first mode, called the new call event mode. The second mode is a general event mode initiated by all SMS and field events except the addition of a new service call. The new call

environment by the assignment of the new call, relative to the stress of the root node 72. Thus, the stress value of the root node 72 can be assumed to be zero. The incremental stress of the field service environment is merely the incremental stress to the particular technician affected by the assignment. Estimation of the stress value by the assigner module 22 involves activation of the scheduler module 24 and will be described in greater detail later in this specification.

The assigner module 22 next sorts the first-level nodes 74, 76 by stress value, and adds them to a search queue for further expansion, as indicated by block 48. The first-level node having a minimum stress value is selected as the "least-stress" node, and is placed first in the search queue. The assigner module 22 then calculates an initial stress threshold based on the minimum stress value, as indicated by block 50. The stress threshold constitutes the sum of the minimum stress value and a user-definable stress margin. This stress threshold is maintained throughout the search until a lower-stress node is generated, at which time the stress threshold is recalculated. The assigner module 22 selects the first node in the search queue for the next expansion, in accordance with the "best-first" characteristic of the search technique which dictates that least-stress nodes be explored first, and executes the next expansion by forming one or more of the next-level nodes 78-84 shown in search tree 70.

Before further expansion, however, the assigner module 22 first determines whether certain search terminating conditions have been satisfied, as indicated by block 52 of Fig. 3. The assigner module 22 expands the first node in the search queue only if the conditions are not satisfied. Although it is unlikely in the real world that the terminating conditions will occur at the first level (I) of the search tree, the assigner module 22 nevertheless checks the conditions at each level of the search. Specifically, the assigner module 22 terminates the search if either the search queue is empty, the stress value of the first, least-stress node is greater than the prevailing stress threshold, the number of nodes generated is greater than a predetermined node count limit, or processing time has exceeded a predetermined time limit. If any of the terminating conditions is met, the assigner module 22 terminates the search and creates an assignment bet corresponding to the best node produced so far, as indicated by branch 54 and block 58 of Fig. 3, and returns the assignment set to a system user as a recommendation.

In the exemplary search tree 70 of Fig. 4, it is clear that the search queue is not empty at the first level, in view of the presence of first-level nodes 74 and 76. In fact, the occurrence of an empty search queue is a rare occurrence at any level in a real world field service environment because the boundaries of the technicians' individual service territories typically overlap such that changes in the technician and call sets can in principle propagate across the entire service territory. As the search progresses, however, there is a possibility that the number of potential next-level nodes may be exhausted, resulting in an empty search queue. Each of the next-level nodes 78-84 corresponds to the respective one of the first-level nodes 74, 76 from which it stems, but is further defined by a reassignment of one of the pending service calls between the selected one of the service technicians, i.e., the service technician to whom the new service call was assigned in the first-level node, and another of the candidate service technicians. Thus, the empty search queue condition occurs only when next-level nodes have been formed for every possible combination of reassignments of the pending service calls between candidate technicians for the respective calls.

Because the search technique is an "anytime" search, the assigner module 22 may terminate the search at any point beyond the root node 72, without affecting the completeness of the assignment and scheduling solution. This "anytime" feature enables the assigner module to apply the other terminating conditions shown in block 52 of Fig. 3. For example, the assigner module 22 expands the first node in the search queue only if it satisfies the prevailing stress threshold. At the first level of the search tree, at least one node will satisfy the stress threshold. Specifically, the stress threshold is equivalent to the stress value of the least-stress, first-level node plus the stress margin. Thus, the least-stress first-level node clearly will be less than the stress threshold. The stress value of the least-stress first-level node serves as a benchmark for the remaining nodes, thereby limiting the search space. The assigner module 22 does not even consider further expansion of first-level nodes that exceed the stress threshold. However, it is possible that first-level nodes that exceed the stress value of the least-stress node may later produce next-level nodes having lower stress values. Therefore, the value of the user-definable stress margin should be large enough to accommodate the "lumpiness" of the search space, as determined empirically by monitoring the actual field service environment over time, but small enough to avoid wasting time on unproductive expansions.

The node count limit and time limit terminating conditions shown in block 52 provide maximum search constraints designed to further prune the search space. The node count limit specifies a maximum number of nodes that can be generated before the assigner module 22 terminates the search. The time limit defines a maximum elapsed processing time allotted for any single search. The node count limit and time limit conditions work particularly well in practice because the assigner module 22 tends to generate its least-stress nodes early in the search. The early generation of low-stress nodes results not only from the

The in-expansion performed by assigner module 22 comprises the formation of one or more next-level nodes that correspond to the respective first-level node, but which are further defined by a reassignment of one of the pending service calls to the selected service technician, i.e., the technician to whom the new service call was assigned in the first-level node, from another of the service technicians. Thus, the selected technician qualifies as a candidate technician for the respective pending call. In-expansions are represented as  $cF \rightarrow T$ , where  $c$  is the pending call number,  $F$  is the technician from whom the pending call is reassigned, and  $T$  is the technician to whom the pending call is to be reassigned.

Assuming that first-level node 74 is the first node in the search queue, examples of next-level nodes formed by in-expansion are next-level nodes 78 and 80. Next-level node 84 would represent an in-expansion of first-level node 76, provided first-level node 76 were the first node in the search queue. First-level node 74 is defined by the assignment of pending calls 1 and 2 to technician A, the assignment of new call 5 to technician B, and the assignment of pending calls 3 and 4 to technician C. As in the out-expansion operation, in-expansion of first-level node 74 does not disturb the assignment of new call 5 to the selected technician B. Rather, in-expansion results in a next-level node 78 that corresponds to first-level node 74, but which is defined by a reassignment of pending call 1 from technician A to the selected technician B. As shown in Fig. 4, the expansion component 74b of first-level node 74 represents the above in-expansion as  $P: 1A \rightarrow B$ .

The assigner module 22 repeatedly invokes the in-expansion operation one or more times to form in-expanded next-level nodes defined by each possible reassignment of pending calls to the selected technician, one call at a time. Similarly, the out-expansion operation is repeatedly invoked to form out-expanded next-level nodes defined by each possible reassignment of the pending service calls assigned to the selected technician. Of course, in each repetition, the reassignment must be directed to a candidate technician for the respective pending call. It is also noted that the assigner module 22 avoids generating the same expansions for a given technician more than once, and therefore does not repeat the reassignment of a pending call on any given path from the root node 72 of search tree 70.

For each of the next-level nodes, the assigner module 22 estimates a stress value representing a degree of undesirability of the respective reassignment of the pending service call, as indicated by block 60 of Fig. 3. The technique by which the assigner module 22 estimates the stress value for each of the next-level nodes is identical to that employed for the first-level nodes, requiring activation of the scheduler module 24, and, for this reason, will be described later in the specification. As indicated by block 62, the assigner module 22 next updates the prevailing stress threshold by determining the minimum stress value of the stress values estimated for each of the next-level nodes. If one of the stress values is less than the minimum stress value determined for the first-level nodes, the assigner module 22 recalculates the stress threshold. For example, assuming that first-level node 74 previously was the least-stress node, but that next-level node 78 generated an even lower stress value, the assigner module 22 updates the stress threshold to reflect the stress value of next-level node 78 plus the user-defined stress margin. However, if none of the next-level nodes has a stress value less than that of the previous least-stress node, the stress threshold is unchanged.

The assigner module 22 limits the depth of expansion by comparing the level of the new set of next-level nodes to a depth limit, as indicated by block 64. The depth limit specifies a maximum depth of the search tree 70. The assigner module 22 does not consider next-level nodes at the depth limit for further expansion, and does not add them to the search queue. Rather, if the level of the new set of next-level nodes is equal to the depth limit, the assigner module 22 carries out the next expansion cycle by expanding only the least-stress unexpanded node already in the search queue, as indicated by loop 66. If the level of the new next-level nodes is less than the depth limit, however, the assigner module 22 sorts them by stress value and merges them into the search queue with the unexpanded nodes, as indicated by block 68. Thus, new next-level nodes at a level less than the depth limit are considered for the next expansion, as indicated by loop 66.

Before starting the next expansion cycle, the assigner module 22 first checks the terminating conditions of block 52. The first node in the search queue now represents the least-stress node of the remaining unexpanded first-level and next-level nodes. At this stage of the expansion, all next-level nodes are unexpanded. With reference to Fig. 4, for example, if first-level node 74 was the previous least-stress node, and node 74 was expanded by forming next-level nodes 78, 80, the nodes in the search queue would comprise unexpanded first-level node 76 and unexpanded next-level nodes 78 and 80. If the terminating conditions are not satisfied, the assigner module 22 starts the new expansion cycle. Specifically, as indicated by block 58, the assigner module 22 removes the first node from the search queue and again executes the expansion operation by forming one or more next-level nodes.

margin. Before executing an expansion cycle, the assigner module 22 checks the terminating conditions of block 118 which correspond to the terminating conditions applied in the new call event mode. The occurrence of one of the terminating conditions results in the termination of the search, as indicated by branch 120 and block 122. At the root level of the search tree, however, the terminating conditions of block 118 will never be satisfied.

The assigner module 22 next removes the root node from the search queue and expands it as a preceding-level node by forming one or more next-level nodes, as indicated by block 124 of Fig. 5. Each of the next-level nodes corresponds to the root node, but is further defined by a reassignment. Specifically, each of the next-level nodes is defined by the reassignment of one of the pending service calls between a selected one of the service technicians and another of the service technicians. After the initial expansion of the root node for the selected technician, the assigner module 22 forms subsequent next-level nodes by propagating potential changes along the paths of the search tree. If the general event is a call attribute change, the selected technician is the technician to whom the changed call is assigned. The changed call may itself be reassigned to more appropriate technicians. If the general event is a technician attribute change, the selected technician is the technician having the changed attributes. If the general event is a request for further evaluation, the selected technician is either one of the technicians for whom the system user requested evaluation, or one of the technicians assigned a call for which the system user requested evaluation. If the general event is the passage of a predetermined amount of time, the selected technician is determined by the status of the pending calls assigned to the technician. This feature is designed to update the assignment set according to changes in the status of service calls over time, such as the completion of a call by a service technician, or to update out-of-date recommendations. The update principally involves operation of the scheduler module 24 to update the existing schedule set. Finally, if the general event involves the cancellation of a call, the selected technician is the technician to whom the cancelled call was previously assigned. In this manner, the assigner module 22 evaluates the reassignment of calls to and from the particular technician, in view of the modified assignment load of the technician.

The next-level nodes result from either an in-expansion or an out-expansion of the root node, as described with respect to the new call event mode. Thus, the operation of the assigner module 22 in the general event mode basically corresponds to the portion of the new call event mode starting with the expansion of the first set of next-level nodes 78-84 shown in the search tree 70 of Fig. 4. After expanding the root node, the assigner module 22 estimates stress values for each of the resulting next-level nodes, representing a degree of undesirability of the reassignment of the respective service call, as indicated by block 126 of Fig. 5. Again, estimation of the stress value requires the assigner module 22 to invoke the scheduler module 24, as will be discussed. The assigner module 22 then sorts the next-level nodes by stress value, and merges the next-level nodes into the search queue, placing the node having the minimum stress value first in the search queue.

As in the new call event mode, the assigner module 22 determines whether the stress-threshold should be updated. If one of the stress values of the next-level nodes is less than the minimum stress value determined for the root node, the assigner module 22 recalculates the stress threshold. The updated stress threshold then represents the sum of the minimum stress value of the next-level nodes and the stress margin. However, if none of the next-level nodes has a stress value less than that of the root node, the stress threshold is unchanged. As indicated by block 130, the assigner module 22 next applies the depth limit to determine whether the new set of next-level nodes should be added to the search queue. If the new nodes satisfy the depth limit, the assigner module 22 sorts them by stress value, and merges them into the search queue, as indicated by block 134. The assigner module 22 otherwise begins the next expansion without the new nodes. As indicated by loop 132, the assigner module 22 then checks the terminating conditions shown in block 118 to determine whether a new expansion should be executed. If the terminating conditions are not met, the assigner module 22 begins a new expansion cycle by removing the first one of the unexpanded next-level nodes in the search queue, and expanding it to form one or more next-level nodes, as indicated by block 124. Thus, the first node in the search queue serves as a preceding-level node for the generation of the next level of the search tree. As in the new call event mode, the assigner module 22 conducts the stress estimation and stress threshold operations of blocks 126 and 128, respectively, and continues to generate new expansion cycles, according to the "best-first" search technique, by in-expansions and out-expansions until the terminating conditions of block 118 are met. The continued expansion results in a succession of next-level nodes. Upon occurrence of one of the terminating conditions, the assigner module 22 generates a new, recommended assignment set corresponding to the node having the minimum stress value.

Estimation of the stress value by the assigner module 22 and scheduler module 24 in both the new call and general event nodes will now be described in detail. The primary goal of A/S system 12 is to minimize

of the service territory of the technician to whom the call is assigned, but high if the location falls outside of an acceptable travel time of the technician's territory. The assigner module 22 compares the service territory of the technician to the location associated with a call by reference to both the technician attributes stored in the technician set data structure 28 and the call attributes stored in the call set data structure 30.

- 5 Incorporation of the territory stress value  $tv$  in the stress function reduces unnecessary travel time for the technicians.

The skill stress value  $kv$  represents a number of skills, or resources, for which the technician to whom a call is assigned is qualified. The skills can be weighted to reflect relative values of individual skills such that the skill stress value  $kv$  represents a sum of the values. The purpose of the skill stress value  $kv$  is to retain the more broadly-trained technicians and technicians trained on higher-priority equipment "in reserve" if less broadly-trained technicians are available for the same service call. The assigner module 22 determines the skill stress value  $kv$  for a technician by reference to the technician attributes stored in the technician set data structure 28.

- 10 The depth stress value  $dv$  represents the number of levels generated in the search tree. This value penalizes prolonged searches, discouraging the continued search for nodes that produce only minimal improvements over nodes already obtained, and preventing excessive geographic spread. The assigner module 22 simply maintains a count of the number of levels generated in the search tree to determine the depth stress value  $dv$ .

The schedule stress value  $sv$  represents the degree of undesirability of the best schedule that can be generated for a particular technician. A schedule is a time-ordered sequence of the calls assigned to a technician. The "best" schedule is the particular sequence of calls producing the lowest schedule stress value. Estimation of the schedule stress value  $sv$  requires operation of the scheduler module 24.

- The assigner module 22 invokes the scheduler module 24 during the search when a call is added to a technician's schedule by assignment of a new call or reassignment of a pending call, a call is removed from a technician's schedule by reassignment, or a request for further evaluation is received. Specifically, the assigner module 22 invokes the scheduler module 24 when the stress value of an assignment or reassignment is to be estimated, to thereby add the schedule stress value  $sv$  estimated by scheduler module 24 to the assignment stress calculation represented by equation (3). Each time the scheduler module 24 is invoked, the assigner module 22 passes a set of calls assigned to a technician who is affected by a particular assignment or reassignment, or is the subject of a request for further evaluation. The assigner module 22 also passes a set of call attributes with the calls, a set of technician attributes for the technician to whom the call is assigned, and a current time as determined by reference to the clock 32. The scheduler module 24 uses the call attributes, technician attributes, and current time in the schedule stress calculation, as will be described. The scheduler module 24 responds to the assigner module 22 by generating a plurality of potential schedules of the calls assigned to the particular technician, and then estimates the schedule stress value of each of the potential schedules. The scheduler module 24 selects the potential schedule having the lowest schedule stress value as the "best" schedule. The lowest schedule stress value then serves as the schedule stress value  $sv$  in the assignment stress calculation (3). In addition, the assigner module 22 adds the "best" schedule to the schedule set defining the particular node under evaluation.

- The existing assignment set includes an existing schedule set that provides, for each of the technicians, an existing schedule of the pending calls assigned to the respective technician. As the search progresses, the assigner module 22 generates a schedule set for each of the first-level nodes and next-level nodes. Thus, the schedule set for each of the first-level nodes corresponds to the existing schedule set but includes a schedule of the new service call and the pending service calls assigned to the selected technician in the respective node. The schedule set for each of the next-level nodes corresponds to the preceding-level node from which the respective node stems, but includes schedules of the pending service calls assigned and reassigned to each of the service technicians involved in the respective reassignment. When the search process is terminated, the assigner module 22 adds to the new, recommended schedule set the resultant schedule set for the particular one of the nodes having the minimum assignment stress value.

- For each of the first-level nodes, for example, the scheduler module 24 effectively modifies the existing schedule set by the addition of a new schedule, which replaces a previous schedule. The scheduler module 24 arrives at the new schedule by generating one or more potential schedules of the calls passed by assigner module 22 for the selected technician. The calls passed by assigner module 22 for a first-level node include the pending calls previously assigned to the selected technician and the new call added to the technician's schedule. The scheduler module 24 estimates a schedule stress value of each potential schedule to determine a relative degree of undesirability of the schedule. The scheduler module 24 then

commitment value is included in the call attributes passed by assigner module 22.

(6)  $\lambda_a$  is a tardiness value for the call of assignment  $a$ . The tardiness value is 0 if the technician to whom the call is assigned is expected to arrive before a promised start time. Otherwise, the tardiness value represents a time interval between the promised start time for the respective service call and an expected start time for the service call. The scheduler module 24 determines the tardiness value by comparing the scheduled time for a call with the promised start time.

(7)  $\omega_N$  is a multiplier for the tardiness value of uncommitted calls set by the field service organization.

(8)  $\omega_C$  is a multiplier for the tardiness value of committed calls set by the field service organization.

(9)  $\theta_a$  is the travel time value representing a time required by the technician to whom the respective service call of assignment  $a$  is assigned to travel to a location associated with the call from a preceding location. The scheduler module 24 may determine the travel time by reference to a travel time file listing estimated times for travel between specific locations. Alternatively, the scheduler module 22 may refer to postal zip codes for the locations associated with each call, as indicated in the call attributes, and calculate the travel time based on the distance between the centroids of the zip codes.

(10)  $\omega_t$  is a multiplier for the travel time value set by the field service organization.

The technique by which the scheduler module 24 generates potential schedules will now be described in greater detail. Because the scheduler module 24 is repeatedly invoked by the assigner module 22 to evaluate assignments, its performance is one of the principle factors in the performance of the entire A/S system 12. Therefore, the amount of processing required for each scheduling operation should be kept to a minimum. In the absence of processing constraints, however, the potential schedules generated by scheduler module 24 could include an extremely high number of possible call sequences for a particular technician. For  $n$  number of calls assigned to the technician, the scheduler module 24 could generate  $n!$  different potential schedules in some cases. To reduce this combinatorial complexity, the scheduler module 24 employs a combination of pruning techniques to reduce the amount of processing required for each scheduling operation. Specifically, the scheduler module 24 employs a pair of pruning heuristics designed to completely avoid the necessity of a full-scale scheduling operation, and a third pruning heuristic designed to reduce the number of potential schedules generated during a scheduling operation. Thus, the scheduler module 24 generates an original potential schedule only as a last resort after application of the pruning heuristics.

The first and second pruning heuristics will be described with reference to Figs. 6 and 7. The first pruning heuristic is herein referred to as "caching," whereby the scheduler module 24 avoids the redundant generation of potential schedules that have already been generated. In response to the addition of a call, the removal of a call, or a request for reevaluation, as indicated by block 140 in the flow diagram of Fig. 6, the scheduler module 24 must generate a least-stress schedule of the set of calls passed by assigner module 22 for a particular technician. Before generating any potential schedules, however, the scheduler module 24 searches a plurality of schedules stored, or "cached," in a schedule buffer associated with the scheduler module 24. The schedules stored in the schedule buffer represent "best" schedules generated in previous runs of the scheduler module 24. The schedule buffer stores each schedule with its corresponding schedule stress value, as determined previously by the scheduler module 24. The scheduler module 24 maintains the contents of the schedule buffer for a predetermined amount of time for future reference. The scheduler module 24 accesses the schedule buffer in an attempt to find a "non-obsolete," matching schedule previously generated for the same set of calls and the same technician passed by the assigner module 22, as indicated by block 142. A schedule is "non-obsolete" if a better schedule has not been generated in a subsequent run of the scheduler module 24, and if a significant amount of time has not passed. The matching schedule represents a time-ordered sequence for an identical assignment of calls to the same technician under evaluation. If a matching schedule is located, the scheduler module 24 simply selects that schedule as the best schedule for the particular technician and returns it to the assigner module 22 with the corresponding schedule stress value, as indicated by branch 144 and block 146. Thus, the caching technique avoids performing the same scheduling work more than once.

If the scheduler module 24 does not locate a matching schedule in the schedule buffer, the scheduling operation advances to the second pruning heuristic. The second pruning heuristic involves the calculation of a lower bound for the addition of a new call or a pending call to an original schedule having one or more calls. As indicated by block 148 of Fig. 6, the scheduler module 24 uses the lower bound to prune the schedules that are too expensive. Specifically, if the addition of the call produces a stress value that is greater than the lower bound, the particular node is deemed too expensive, and the scheduler module 24 notifies the assigner module 22 that the node should be discarded without further evaluation. The scheduler module 24 notifies the assigner module 22 by returning an indication that the scheduler module 24 failed to generate a schedule within acceptable stress limits, as indicated by branch 150 and block 154.

If the lower bound  $y$  is not less than the worst acceptable schedule stress value  $\psi_s$ , the scheduler module 24 indicates to the assigner module 22 that the node is to be discarded. If the lower bound is satisfied, however, the scheduler module 24 advances to the next stage in the scheduling operation.

If the caching technique does not produce a matching schedule, and the lower bound test is satisfied, the scheduler module 24 must produce a minimum stress schedule for the set of calls passed by the assigner module 22, as indicated by block 148 of Fig. 6. At this stage, however, the scheduler module 24 applies a third pruning heuristic to reduce the factorial complexity of the scheduling possibilities. Specifically, the scheduler module 24 employs a recursive "depth-first" schedule search technique in which one or more of the calls are prioritized as *critical* calls at each level of recursion. The scheduler module 24 recursively builds a sequence of the calls one-call-at-a-time, but prunes the number of possibilities at each level of recursion by considering placement of only the critical calls next in the schedule sequence. The pruning operation thereby eliminates the need to evaluate the placement of non-critical calls next in the sequence.

One example of a critical call is a committed call, where an arrival time has been promised to the customer. Because the committed call should not be scheduled later than the promised arrival time, it is designated a critical call that must be considered for placement next in the schedule sequence. Thus, initially the scheduler module 24 determines whether a call  $a$  is critical based on the definition

$$critical = \{a: a \text{ has } C = 1\} \quad (11)$$

where  $C = 1$  indicates that the call  $a$  has been committed. The critical status of a committed call is maintained throughout the recursion by scheduler module 24.

If the definition yields  $critical = \emptyset$ , however, the scheduler module 24 determines whether calls are critical based on their travel times  $\theta_n$  from the particular technician's starting location and tardiness  $\lambda_n$ , as well as by their priorities  $P_n$ . At the root of each recursion, the technician's starting location is the technician's current location. As the recursion progresses to build a schedule sequence, however, the starting location is the location associated with the immediately preceding call in the sequence. This change in the starting location requires the recalculation of *critical* at each level of the recursion. Consequently, the calls designated as *critical* will change as the recursion progresses.

Each call is evaluated for *critical* based on the following tests:

(A) Location: Is the travel time less than or equal to the average travel time for this set of calls,

$$\theta_a \leq \frac{\max(\theta_n) + \min(\theta_n)}{2} ?$$

(B) Tardiness: Is tardiness greater than or equal to the average tardiness for this set of calls,

$$\lambda_a \geq \frac{\max(\lambda_n) + \min(\lambda_n)}{2} ?$$

(C) Priority: Is this a priority call,

$$P_a > 0?$$

The scheduler module 24 creates the following partitions based on tests (A)-(C):

$\alpha_3$ : This partition contains the calls meeting all three tests (A)-(C);

$\alpha_2$ : If the number of calls in  $\alpha_3 \geq 1/2$  of the number of calls in this schedule,  $\alpha_2 = \emptyset$ ; else,  $\alpha_2$  contains all calls that meet any two of tests (A)-(C); and

$\alpha_1$ : If the number of calls in  $\alpha_3 + \alpha_2 \geq 1/2$  of the number of calls in this schedule,  $\alpha_1 = \emptyset$ , else  $\alpha_1$  contains all calls that meet any of tests (A)-(C).

The set of *critical* calls is thus:  $critical = \{\alpha_3 \cup \alpha_2 \cup \alpha_1\}$ . The union ensures that a sufficient number of calls will be designated as *critical* calls at each stage of the recursion, while the partitions ensure that the number of critical calls is minimized.

time blocks specified by the technician calendar stored in the technician attributes passed by assigner module 22. The available time segments generally will correspond to a maximum of an entire work day, which may run, for example, from 8:00 to 17:00. The technician's calendar tracks appointments and other times unavailable for the scheduling of calls as fixed blocks of time at points during the time segment specified by the organizational calendar. The unavailable times are associated with locations that are used to calculate travel time between the appointment and service calls. The available time segments are hashed into time bins, with the beginning of each bin being either the start of the day, or the end of the previously scheduled call or unavailable time. The end of a time bin is then either the end of the same day, or the start of either the next scheduled call or an unavailable time. The scheduler module 24 caches the set of time bins for each technician to avoid regeneration at each level of the schedule search recursion.

Fig. 9 is an example of a technician's schedule divided into time segments and time bins. Each of the time segments 200, 202, 204 corresponds to an entire work day, Tuesday, Wednesday, Thursday, respectively. The first time segment 200, Tuesday, includes an unavailable block of time 206 from 08:00 to 14:00, and an available time bin 208 beginning at the end of block 206, plus travel time 212. Thus, the scheduler module 24 also considers travel times 210, 212, and 214 as part of the time segment 200. The time bin 208 ends at 17:00, the end of the time segment, minus the travel time 214. The Wednesday time segment 202 does not include any unavailable block of time, and therefore consists of a single time bin 216 spanning the entire work day, but bordered by travel times 218, 220. In contrast, the Thursday time segment 204 is hashed into two time bins 222, 224. The first time bin 222 begins at the beginning of the work day, 8:00, plus travel time 226, and ends at the beginning of an unavailable block of time 228, minus travel time 230. The second time bin 224 begins at the end of block 228, plus travel time 232, and runs through the end of the work day, 17:00, minus travel time 234.

When the scheduler module 24 selects a call to be next in the sequence of a potential schedule, it determines a start time for the call based on the following factors:

- (1) a travel time  $\theta_{n \rightarrow m}$  between locations associated with a preceding call  $n$  and the next call  $m$  to be scheduled, or between locations associated with the next call  $m$  and either a preceding block of unavailable time or the beginning of the time segment;
- (2) a duration  $\rho_m$  of the next call  $m$  to be scheduled based on an expected repair time of the machine associated with the call  $m$ ; and
- (3) a length  $\eta_{bin}$  of the time bin in which the next call  $m$  is to be scheduled.

Thus, to ensure that the duration of the next call  $m$  fits within the time bin, indicating available scheduling time, the scheduler module 24 schedules call  $m$  next in the sequence only if:

$$\theta_{n \rightarrow m} + \rho_m + \theta_{c \rightarrow ?} \leq \eta_{bin}$$

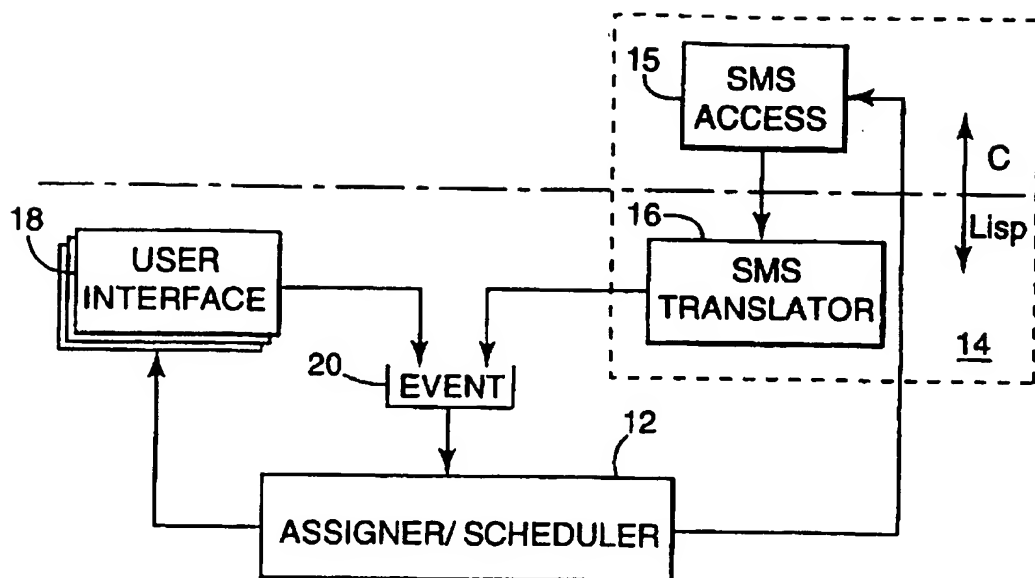
where  $\theta_{c \rightarrow ?}$  represents the travel time between locations associated with the call  $m$  to be scheduled and the next call to be scheduled after call  $m$ .

Fig. 10 is an example of a graphical representation of a portion of the assignment set and schedule set generated by A/S system 12, as displayed to the system user by user interface 18. The user interface 18 preferably displays an interactive scheduler window 240 containing a representation of the calls assigned to each technician and the times at which the calls are scheduled. The scheduler window 240 includes a technician field 242 containing a representation of a particular group of technicians under evaluation by the system user, a schedule field 244 containing a representation of the calls assigned to each of the technicians and the particular times for which the calls are scheduled, and a command bar 246 containing representations of standard window control commands.

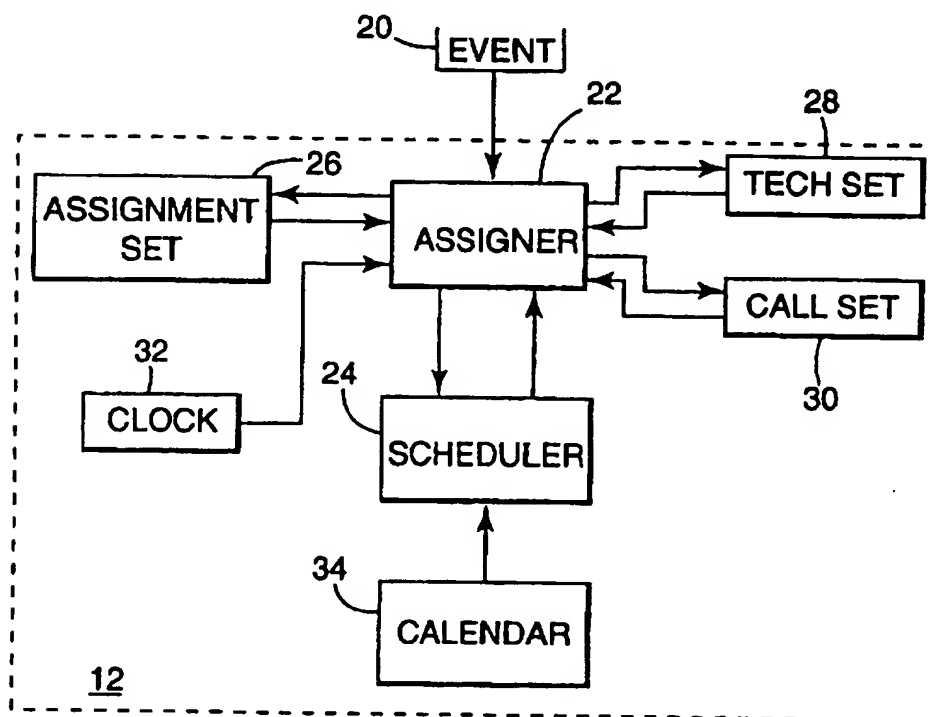
In Fig. 10, the technician field 242 displays a group of technicians A, B, C, D, and E operating in the field service environment. The schedule field 244 represents the existing schedules of the technicians, as generated by the scheduler module 24, subject to approval or modification by the system user. The schedule field 244 in Fig. 11 indicates that technician A has been assigned first, second, and third scheduled calls represented by call blocks 248, 250, and 252, which define assignments. The call blocks 248, 250, 252 represent scheduling of the calls between 9:00 and 15:00 in a time segment corresponding to Monday, May 17. Specifically, call blocks 248, 250, and 252 indicate assigned start times for the first, second, and third calls of approximately 9:00, 11:15, and 1:15, respectively. The completion times indicated by call blocks 248, 250, 252, respectively, are approximately 10:45, 12:45, and 2:45.

The schedule field 244 also includes time blocks 254, 256, 258 representing travel times. Time block 254 indicates the initial travel time, from the beginning of the day, required for the technician to travel from the starting location, or from a location associated with an otherwise unavailable time, to the customer location associated with the first call. Time blocks 256 and 258 represent the travel times that separate the

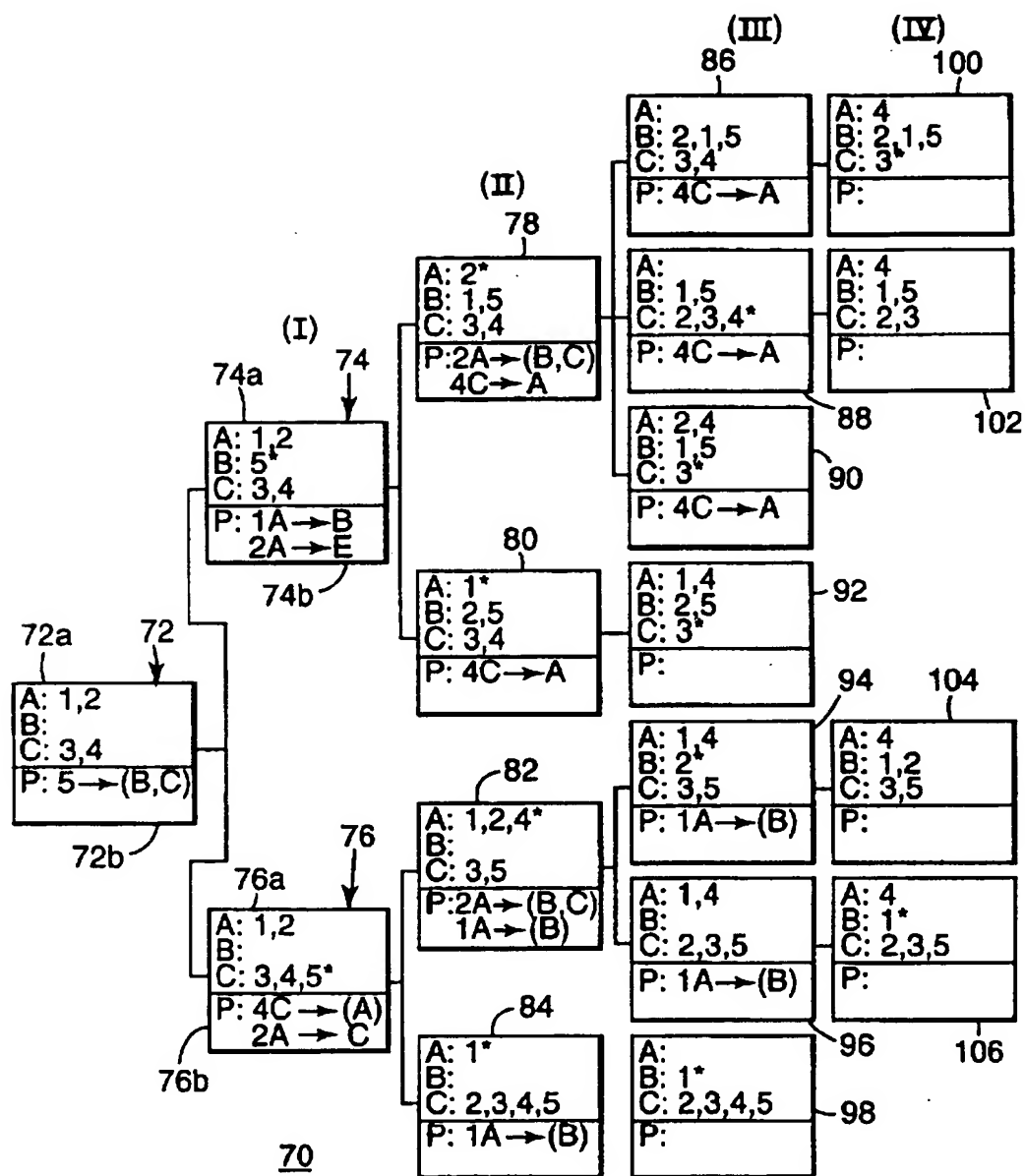
4. The method according to claim 3, characterized in that said existing assignment set includes an existing schedule set, said existing schedule set including, for each of said resource providers, an existing schedule of the pending resource requests assigned to the respective resource provider, said method further comprising the steps of:
  - 5 generating, for each of said first-level nodes, a schedule set corresponding to said existing schedule set but including a schedule of said new resource request and the pending resource requests assigned to said selected one of said resource providers, and
  - generating, for each of said next-level nodes, a schedule set corresponding to the schedule set for the respective preceding-level node but including schedules of the resource requests assigned and
  - 10 reassigned to said one and said another of said resource providers,
  - wherein said new assignment set includes the schedule set for the respective one of said first-level nodes and said next-level nodes having said minimum stress value.
5. The method according to claim 4, characterized in that said step of generating a schedule set for each of said first-level nodes includes the steps of:
  - 15 generating one or more potential schedules of said new resource request and the pending resource requests assigned to said selected one of said resource providers, estimating, for each of said potential schedules, a schedule stress value representing a degree of undesirability of the respective potential schedule, and
  - 20 selecting one of said potential schedules having a minimum schedule stress value as said schedule for said selected one of said resource providers.
6. The method according to claim 5, characterized in that said step of generating a schedule set for each of said next-level nodes includes the steps of:
  - 25 generating one or more potential schedules, said potential schedules including one or more first potential schedules of the resource requests assigned to said one of said resource providers, and one or more second potential schedules of the resource requests assigned to said another of said resource providers,
  - estimating, for each of said potential schedules, a schedule stress value representing a degree of
  - 30 undesirability of the respective potential schedule, and
  - selecting one of said first potential schedules having a first minimum schedule stress value as said schedule for said one of said resource providers, and one of said second potential schedules having a second minimum schedule stress value as said schedule for said another of said resource providers.
- 35 7. The method according to claim 6, characterized in that, in said step (a)(ii), the stress value estimated for each of said first-level nodes includes a representation of said minimum schedule stress value for the respective first-level node, and, in said steps (b) and (f), the stress value estimated for each of said next-level nodes includes a representation of said first minimum schedule stress value and said second minimum schedule stress value for the respective next-level node.
- 40 8. The method according to any one of claims 5 to 7, characterized in that said step of generating said one or more potential schedules for each of said first-level nodes includes, before generating said potential schedules, the steps of:
  - estimating a worst acceptable schedule stress value,
  - 45 estimating, for each of said first-level nodes, a minimum stress of insertion of said new resource request among the pending resource requests on the existing schedule for said selected technician, and
  - discarding each of said first-level nodes having a minimum stress of insertion that exceeds said worst acceptable schedule stress value.
- 50 9. The method according to claim 6 or 7, characterized in that said step of generating said one or more first and second potential schedules for each of said next-level nodes includes, before generating said potential schedules, the steps of:
  - estimating a worst acceptable schedule stress value,
  - 55 estimating, for each of said next-level nodes, a minimum stress of insertion of the pending resource request reassigned to one of said one and said another of said resource providers among the pending resource requests on the existing schedules for the respective one of said one and said another of said resource providers, and



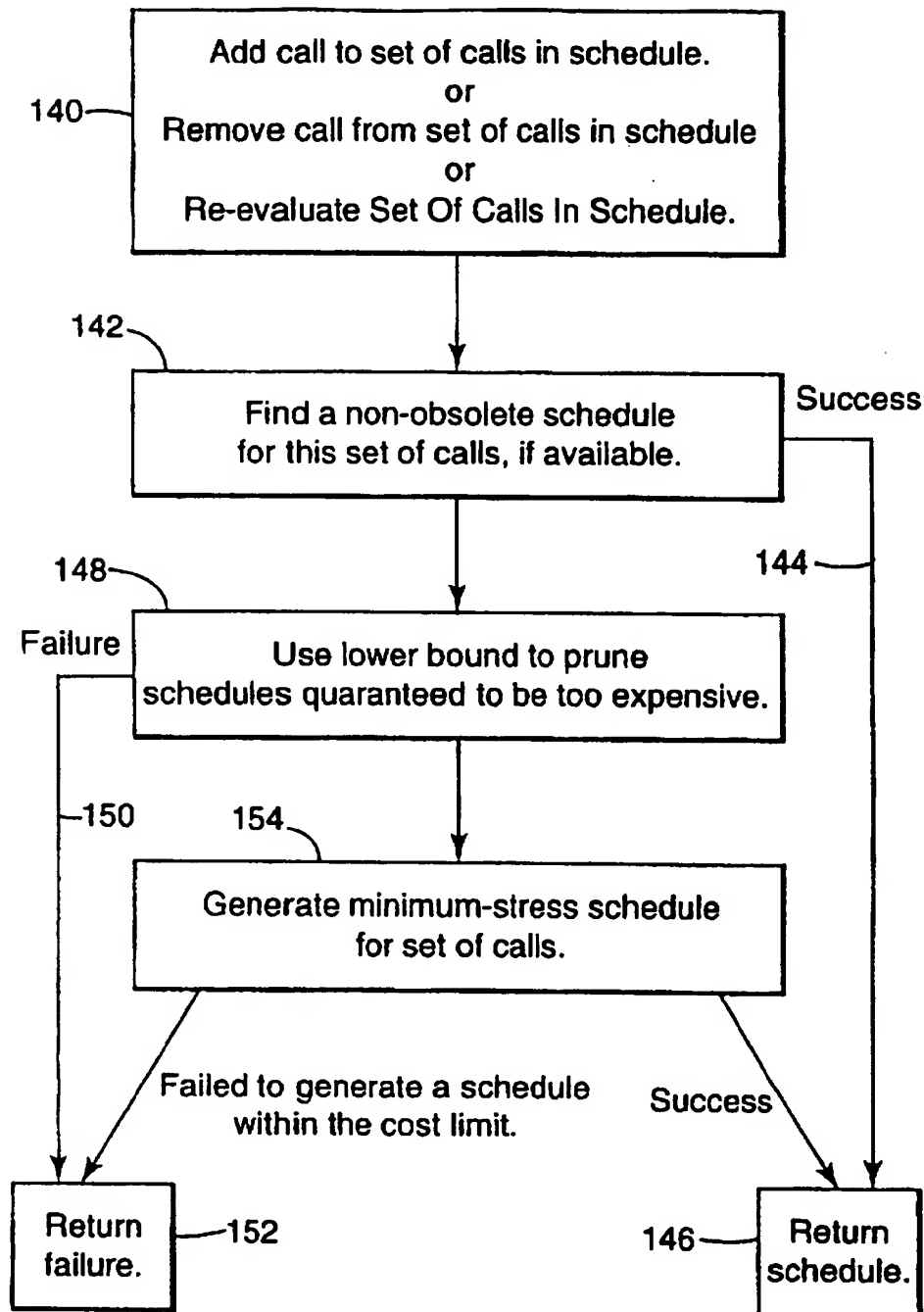
**Fig. 1**



**Fig. 2**



**Fig. 4**

**Fig. 6**

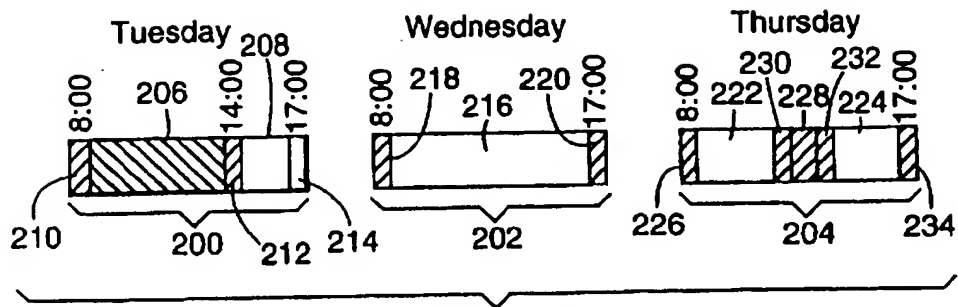


Fig. 9

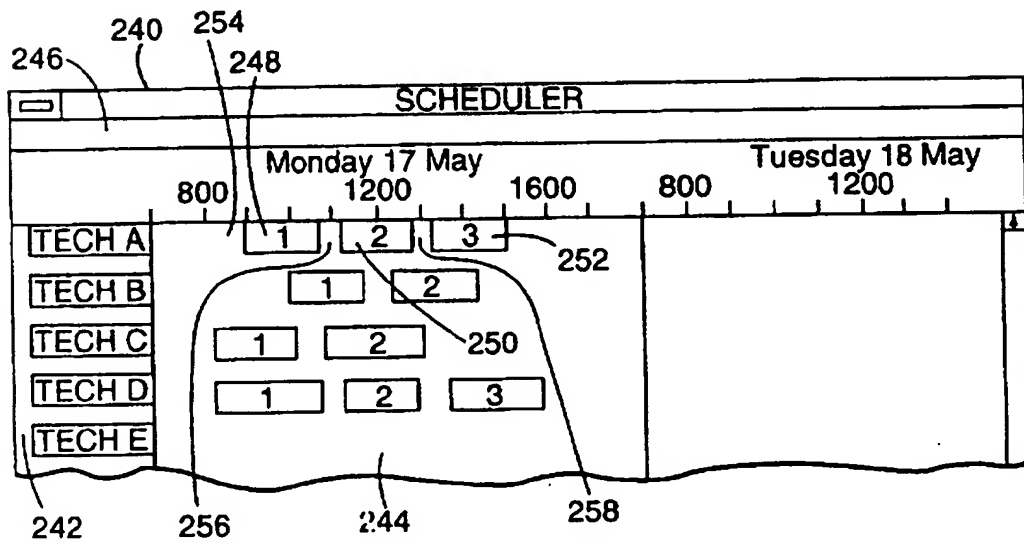
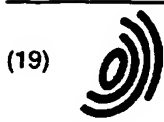


Fig. 10



(19)

Europäisches Patentamt

European Patent Office

Office européen des brevets



(11)

EP 0 669 586 A3

(12)

## EUROPEAN PATENT APPLICATION

(88) Date of publication A3:  
17.04.1996 Bulletin 1996/16

(51) Int. Cl.<sup>6</sup>: G06F 17/60

(43) Date of publication A2:  
30.08.1995 Bulletin 1995/35

(21) Application number: 95102556.8

(22) Date of filing: 23.02.1995

(84) Designated Contracting States:  
DE FR GB IT

(30) Priority: 25.02.1994 US 201664

(71) Applicant: MINNESOTA MINING AND  
MANUFACTURING COMPANY  
St. Paul, Minnesota 55133-3427 (US)

(72) Inventors:  
• Sisley, Elizabeth M.,  
c/o Minnesota Mining and  
Saint Paul, Minnesota 55133-3427 (US)

• Collins, John E.,  
c/o Minnesota Mining and  
Saint Paul, Minnesota 55133-3427 (US)

(74) Representative: Hilleringmann, Jochen, Dipl.-Ing.  
et al  
Patentanwälte  
von Kreisler-Setling-Werner,  
Bahnhofsvorplatz 1 (Deichmannhaus)  
D-50667 Köln (DE)

(54) System and method for resource assignment and scheduling

(57) A system and method for assigning and scheduling resource requests to resource providers use a modified "best-first" search technique that combines optimization, artificial intelligence, and constraint-processing to arrive at near-optimal assignment and scheduling solutions. In response to changes in a dynamic resource environment, potential changes to an existing assignment set are evaluated in a search for a better solution. New calls are assigned and scheduled as they are received, and the assignment set is readjusted as the field service environment changes, resulting in global optimization. Each search operation is in response to either an incremental change to the assignment set such as adding a new resource request, removing a pending resource request, reassigning a pending resource request, or to a request for further evaluation. Thus, the search technique assumes that the existing assignment set is already optimized, and limits the task only to evaluating the effects of the incremental change. In addition, each search operation produces a complete assignment and scheduling solution. Consequently, the search can be terminated to accept the best solution generated so far, making the technique an "anytime" search.

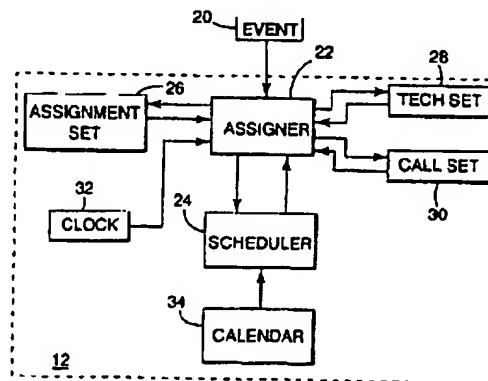


Fig. 2

EP 0 669 586 A3